Amendments to the Claims:

This listing of claims will replace all prior versions and listings of claims in the application.

1. (currently amended) A method for mapping a descriptive language including a data description having a structure complexity into an object oriented data presentation, comprising the steps of:

~~identifying~~ receiving the data description;

identifying a complex-type element in the data description; and

creating an executable object oriented class corresponding to the identified complex-type element, wherein the class includes ~~including~~ an internal static class, wherein the internal static class corresponds to the structure complexity of the data description.

2. (currently amended) The method as recited in claim 1, wherein ~~said method further comprises~~ receiving the data description comprises receiving an XML Schema ~~for an XML text~~.

3. (currently amended) The method as recited in claim 1, ~~wherein said identifying includes~~ further comprising validating ~~a Schema~~the data description ~~including a class description to provide the creation of an instance of a compiler class corresponding to the class description~~.

4. (currently amended) The method as recited in claim 3, wherein said validating further includes using a object finite state machine including a current state to verify a mutator method call against the current state of the object, wherein the ~~Schema~~ data description is invalid

when the mutator method call is initiated before the current state is complete.

5. (currently amended) The method as recited in claim 3, wherein said validating includes:

sending a request including said ~~Schema~~ data description ~~from a user~~ to a remote server; and

~~retrieving~~ receiving a validity determination as to said ~~Schema~~data description.

6. (currently amended) The method as recited in claim 3, wherein said validating includes:

reading said ~~Schema~~ data description into a set of valid ~~Schema~~ descriptor classes; and

creating a set of objects out of the ~~Schema~~ data description wherein the occurrence of an object reflects validity.

7. The method as recited in claim 1, wherein said creating includes a set of object oriented classes selected from the group consisting of: Java, C++ and Smalltalk.

8. The method as recited in claim 1, wherein said creating includes representing a naming space with said internal static class to provide an implementation of said structure complexity.

9. (currently amended) A method for mapping a Schema including a structural complexity into an executable object oriented language, wherein the object oriented language

~~including a functionality to provide~~ provides a one to one correspondence between the structural complexity of the ~~semantical language~~ Schema and the functionality of the object oriented language, comprising the steps of:

receiving said Schema;

validating said Schema;

creating a set of object oriented classes including a set of internal static classes to provide a mapping of the Schema into the object oriented language;

creating an instance corresponding to the object oriented classes;

compiling the instance to provide an object oriented code~~; and~~

~~transmitting the object oriented code~~.

10. The method as recited in claim 9, wherein said validating includes using a object finite state machine including a current state to verify a function call against the current state of the object, wherein the Schema is invalid when the function call is initiated before the current state is complete.

11. (currently amended) The method as recited in claim 9, wherein said validating includes:

sending a request including said Schema ~~from a user~~ to a remote server; and

~~retrieving~~ receiving a validity determination as to said Schema.

12. (currently amended) The method as recited in claim 9, wherein said validating includes:

reading said Schema into a set of valid ~~Schema~~ descriptor classes; and

creating an instance of a compiler class wherein the compiler class is described in the Schema.

13. (currently amended) The method as recited in claim 9, wherein ~~said creating a~~ the set of object oriented classes ~~includes~~ comprises code selected from the group consisting of: Java, C ++, and Smalltalk.

14. The method as recited in claim 9, wherein said creating an instance includes representing a naming space with the internal static class to provide an implementation of said structural complexity.

15. (currently amended) A computer readable medium containing programming which when executed performs the following ~~procedures comprising~~:

~~identifying~~ receiving a data description; ~~and~~

identifying a complex-type element in the data description; and

creating an executable object oriented class corresponding to the identified complex-type element, wherein the class includes ~~including~~ an internal static class, wherein the internal static class corresponds to a structure complexity of the data description.

16. (currently amended) The medium as recited in claim 15, wherein ~~said medium further performs the procedure~~ receiving the data description comprises receiving an XML Schema ~~for an XML text~~.

17. (currently amended) The medium as recited in claim 15, ~~wherein said identifying procedure includes~~ further comprising validating ~~a Schema~~ the data description~~including a class description to provide the creation of an instance of a compiler class corresponding to the class description~~.

18. (currently amended) The medium as recited in claim 17, wherein said validating procedure further includes using a object finite state machine including a current state to verify a mutator method call against the current state of the object, wherein the ~~Schema~~ data description is invalid when the mutator method call is initiated before the current state is complete.

19. (currently amended) The medium as recited in claim 17, wherein said validating procedure includes:

sending a request including said ~~Schema~~ data description ~~from a user~~ to a remote server; and

~~retrieving~~ receiving a validity determination as to said ~~Schema~~data description.

20. (currently amended) The medium as recited in claim 17, wherein said validating procedure includes:

reading said ~~Schema~~ data description into a set of valid ~~Schema~~ descriptor classes; and

creating a set of objects out of the ~~Schema~~ data description wherein the occurrence of an object reflects validity.

21. The medium as recited in claim 15, wherein said creating procedure includes a set of object oriented classes selected from the group consisting of: Java, C ++ and Smalltalk.

22. The medium as recited in claim 15, wherein said creating procedure includes representing a naming space with said internal static class to provide an implementation of said structure complexity.

23. (currently amended) A computer readable medium containing programming for mapping a Schema including a structural complexity into an executable object oriented language, wherein the object oriented language provides ~~including a functionality to provide~~ a one to one correspondence between the structural complexity of the ~~semantical language~~ Schema and the functionality of the object oriented language which when executed performs the following procedures comprising: .

receiving said Schema;

validating said Schema;

creating a set of object oriented classes including a set of internal static classes to provide a mapping of the Schema into the object oriented language;

creating an instance corresponding to the object oriented classes;

compiling the instance to provide an object oriented code.~~; and~~

~~transmitting the object oriented code.~~

24. (currently amended) The medium as recited in claim 23, wherein said validating procedure includes using an object finite state machine including a current state to verify a

function call against the current state of the object, wherein the Schema is invalid when the function call is initiated before the current state is complete.

25. (currently amended) The medium as recited in claim 23, wherein said validating procedure includes:

sending a request including said Schema ~~from a user~~ to a remote server; and

~~retrieving~~ receiving a validity determination as to said Schema.

26. (currently amended) The medium as recited in claim 23, wherein said validating procedure includes:

reading said Schema into a set of valid ~~Schema~~ descriptor classes; and

creating an instance of a compiler class wherein the compiler class is described in the Schema.

27. (currently amended) The medium as recited in claim 23, wherein ~~said creating a~~ the set of object oriented classes ~~procedure includes an object oriented language~~ comprises code selected from the group consisting of: Java, C ++, and Smalltalk.

28. The medium as recited in claim 23, wherein said creating an instance procedure includes representing a naming space with the internal static class to provide an implementation of said structural complexity.

29. (currently amended) An apparatus for mapping a descriptive language including a

data description having a structure complexity into an object oriented data presentation comprising:

means for ~~identifying~~ receiving the data description; ~~and~~

means for identifying a complex-type element in the data description; and

means for creating an executable object oriented class ~~including~~ corresponding to the identified complex-type element, wherein the class includes an internal static class, wherein the internal static class corresponds to the structure complexity of the data description.

30. (currently amended) The apparatus as recited in claim 29, wherein said ~~apparatus further comprises means for receiving a~~ data description comprises an XML Schema ~~for an XML text~~.

31. (currently amended) The apparatus as recited in claim 29, ~~wherein said identifying means includes~~ further comprising means for validating a ~~Schema~~ data description~~including a class description to provide the creation of an instance of a compiler class corresponding to the class description~~.

32. (currently amended) The apparatus as recited in claim 29, wherein said validating means further includes a object finite state machine including a current state to verify a mutator method call against the current state of the object, wherein the ~~Schema~~ data description is invalid when the mutator method call is initiated before the current state is complete.

33. (currently amended) The apparatus as recited in claim 29, wherein said validating

means includes a web browser operable to send a request including said ~~Schema~~ data description ~~from a user computer~~ to a remote server for validation~~, in response to said request, said Schema being validated~~.

34. (currently amended) The apparatus as recited in claim 29, wherein said validating means includes:

means for reading said ~~Schema~~ data description into a set of valid ~~Schema~~ descriptor classes; and

means for creating a set of objects out of the ~~Schema~~ data description wherein the occurrence of an object reflects validity.

35. (currently amended) The apparatus as recited in claim 29, wherein said ~~creating means includes a set of~~ object oriented classes comprise code selected from the group consisting of: Java, C++, and Smalltalk.

36. The apparatus as recited in claim 29, wherein said creating means includes means for representing a naming space with said internal static class to provide an implementation of said structure complexity.